"Method   and   system   for   network   intrusion
detection,   related   network   and   computer   program
product"

5   Field of the invention

The present invention relates to network security
and,   more   specifically,   to   techniques   for   network
intrusion detection.

10

Description of the related art

Network   security   systems   -   such   as   intrusion
detection   systems   (IDS),   intrusion   prevention   systems
15   (IPS)   and   firewalls   -   monitor   network   traffic   on   a
network   with   which   they   are   associated   in   order   to
detect   behavior   that   is   illegal,   malicious,   in
violation   of   policy,   or   that   otherwise   constitutes   a
misuse of the monitored network. The detection of such
20   behavior   constitutes   a   security   event   and   may   trigger   a
variety   of   response   tasks,   such   as   logging   of   the
traffic   that   caused   the   event,   alerting   a   human
operator   of   the   event,   or   disrupting   further   traffic
related to the event with a variety of methods.
25   A   widely   used   method   of   recognizing   misuse   in
network   traffic   consists   in   comparing   the   observed
traffic   with   a   set   of   "misuse   signatures",   or   "attack
signatures"   that   identify   a   set   of   known   misbehaviors,
such   as   network   attack   attempts,   network   policy
30   violations, or network protocol violations.
Signatures   in   a   misuse-based   network   IDS   mostly
aim   at   detecting   misbehavior   specific   to   a   given
network   protocol.   IDSs   may   also   perform   a   variety   of
analysis   tasks,   such   as   decoding,   normalization,   or
35   reassembly   of   the   data   flow,   which   tasks   are   also

2

dependent on the network protocol of the communication.

Most application layer protocols may usually be identified by the use of reserved or standard TCP/UDP port numbers, the detection engine of a conventional IDS may identify the protocol of a given data flow from the TCP/UDP port numbers involved in the flow. TCP and UDP are well known acronyms for Transmission Control Protocol and User Datagram Protocol, respectively.

A disadvantage of this solution is that it makes the assumption that server applications are deployed only at the port numbers that are standard for the application's underlying protocol.

There are several reasons why this assumption may not be correct. A server application may be deployed at a non-standard port because the standard port number is already in use by another server, or to bypass a firewall between the server and some of its potential users, or for other technical reasons. A server application may be deployed at a non-standard port to "hide" it, either for legitimate purposes, to achieve "security by obscurity", or for malicious purposes, to camouflage unauthorized deployment.

Results described in Zhang, Y. and Paxson, V. (2000a) "Detecting Backdoors" in Proc. of the 9th USENIX Security Symposium confirm that, in a production environment, a high number of legitimate servers are often found "listening" on non-standard ports.

The deployment of a given server application at a non standard port may or may not itself constitute a policy violation or misuse of the network. In either case, it may open vulnerabilities in the monitored network, and attacks exploiting these vulnerabilities may evade detection by the IDS coupled to the network. This is because the IDS may not correctly identify the protocol of the data flow containing the malicious

data. In other words, an application listening on a non
standard port may cause false negatives, that is,
misbehavior that goes undetected, for a conventional
network IDS.

5   Protocol identification in a network IDS based on
reserved or standard TCP/UDP port numbers suffers from
this disadvantage. For that reason a conventional
network IDS may allow the security administrator to
configure the detection engine. This may occur by

10   specifying for a given protocol one or more ports that
are associated with that protocol. In that way, a
security administrator can manually decide what
protocols are to be associated with each port and,
consequently, what analysis tasks are to be performed

15   and what signatures are tested against a given data
flow.
          A disadvantage of this method lies in that the
effectiveness of a manual configuration is limited by
the administrator's knowledge of the network, which may

20   be incomplete, and by the highly dynamic nature of most
networks, which would require constant update of the
configuration.
          Conventional network intrusion detection systems
may make use of network discovery capabilities, such as

25   active scanning, vulnerability analysis, or passive
network mapping. Information thus gathered may be used
to reduce the ambiguity inherent in being a passive
observer of data flows, and thus improve the quality of
detection. As an example, an attack attempt effective

30   against a given operating system may be correlated with
the operating system known to be present on the host
that is the target of the attack to obtain more
reliable information on the security event.
          US-B-6 415 321 B1 discloses a domain mapping

35   method and system based on the use of a storage device

4

for storing information on a network domain, acquired
with a variety of methods, including passive capture of
the network traffic, and available on the network by
use of a query engine for a variety of network devices,
5   including intrusion detection systems. The network
information gathered is used to configure the IDS. This
information includes operating system, service, and
vulnerability information.

US-B-6 499 107 discloses a method and system for
10   adaptive network security using intelligent packet
analysis. This prior art arrangement uses the network
information gathered to prioritize the activation of
analysis tasks and of misuse signatures on a network
IDS, in order to provide a more predictable behavior
15   under overload.

Finally, WO-A-02/061510 discloses a port profiling
system adapted to detect unauthorized use of network
assets based on the service port numbers involved in
communication on the network.
20

Object and summary of the invention

The object of the present invention is thus to
enhance network intrusion detection systems especially
25   in terms of flexibility and the capability of detecting
network intrusion in a manner that is rendered
thoroughly independent of the physical configuration of
the network.

According to the present invention, that object is
30   achieved by means of a method having the features set
forth in the claims that follow. The invention also
relates to a corresponding system, a related network as
well as a related computer program product, loadable in
the memory of at least one computer and including
35   software code portions for performing the steps of the

5

method of the invention when the product is run on a computer. As used herein, reference to such a computer program product is intended to be equivalent to reference to a computer-readable medium containing instructions for controlling a computer system to coordinate the performance of the method of the invention. Reference to "at least one computer" is evidently intended to highlight the possibility for the present invention to be implemented in a distributed/ modular fashion.

A preferred embodiment of the invention provides intrusion detection in a network wherein data flows are exchanged using associated network ports and application layer protocols; this is essentially done by monitoring the data flows in the network, detecting information on the application layer protocols involved in said monitored data flows, and providing intrusion detection based on the information on application layer protocols detected. Intrusion detection is thus provided independently of any predefined association between the network ports and the application layer protocols.

Preferably, the arrangement described herein makes use of information on the application layer protocol involved in a given data flow, to decide in real time the appropriate behavior to detect misuse within the data flow, regardless of TCP/UDP ports involved in the communication. Protocol identification is performed by means of passive observation of network traffic, using signature-matching techniques.

In the arrangement described herein, network information is used in real time by the IDS to identify the protocol involved in a given data flow and therefore to correctly analyze the data flow. Furthermore, the arrangement described herein

6

introduces novel methods for performing passive
protocol identification, and an efficient technique
that permits integration of protocol identification and
intrusion detection in a single device.

5   Preferably, analysis tasks and misuse signatures
are not prioritized, nor are they individually enabled
and disabled. Rather, a set of analysis tasks and
misuse signatures is selected that is appropriate for
the protocol of the data flow, amongst a plurality of
10   predefined sets. An advantage of the arrangement
described herein is that each signature set does not
change as a consequence of new network information, so
that processing may be performed once and for all to
construct data structures needed for optimized
15   signature matching algorithms such as set wise string
matching and decision trees.

        Preferably, network information is obtained by
passive observation of network traffic. Such a "passive
scanning" approach provides several advantages over
20   active scanning. Passive scanning is entirely
unobtrusive, cannot be detected, and provides real time
information on any device that is active in the
network. Conversely, active scanning is noisy, consumes
bandwidth, may disrupt normal network operation, is not
25   real time, and may fail to detect network devices that
do not reply to the scanning stimuli, such as low
uptime devices, devices protected by firewalls, or
devices requiring specific stimuli, not available to
the vulnerability assessment system, to elicit a
30   response.

        Still preferably, the protocol of a given data
flow is detected regardless of the port numbers
involved in the communication; the use of the
information thus gathered is not limited to alerting
35   for out of policy events, but also includes usage

within a signature based network IDS.

In the exemplary arrangement described herein, protocol identification is performed based on signature matching and on flow characterization and classification. Real time information on the services deployed in the monitored network, is thus provided regardless of the TCP or UDP port numbers on which these services are deployed. A map of the services deployed on the network is thus constructed, and policy violations or anomalous events generate a security event. Misuse based network intrusion detection is thus enhanced by accurate, real time knowledge of the application layer protocol involved in a given data flow. The set of analysis tasks to perform and the set of misuse signatures to match against the flow are thus identified making it possible to detect misuse attempts directed against a given port without prior knowledge of the application layer protocol listening on the port.

In the exemplary arrangement described herein, protocol identification is performed on a device coupled to the network by observing a given data flow for behavior specific to known protocols. A set of signatures is used, specified in a language similar to a signature language used to specify misuse signatures in a network IDS. As an addition to an ordinary IDS signature language, each signature specifies the protocol that is detected if the signature is triggered. Each signature attempts to match a pattern that is unique to a given protocol, and at the same time is frequently used in that protocol. For that purpose, signatures identifying behavior frequently present in server responses may be used, or signatures identifying common client request-server reply behavior. Signatures exclusively matching a pattern in

client behavior are not used: in fact, they could be
triggered erroneously or maliciously by users outside
the monitored network (see e.g. PCT patent application
03WO-IT00505).

5   In another preferred embodiment of the invention,
the protocol identification method above is refined by
flow characterization and classification of data flows
relative to each server application deployed on the
network. A limited set of features such as packet size,
10   packet arrival times, TCP flags and other header
information are gathered and statistics thereof are
computed.   The   flow   classification   engine   then
classifies the data flows and services into a number of
flow classes. Flow classification performed without
15   payload   content   inspection   is   known   to   permit
discrimination between interactive and non-interactive
traffic, and even identification of specific protocols
such as ssh, or web traffic. The information thus
obtained   may   be   correlated   with   the   information
20   gathered by signature based protocol identification to
provide more reliable identification. The signature
language employed for protocol identification may thus
be enriched to allow testing of flow and service
characteristics obtained by the flow classifier engine
25   above.
       Preferably, protocol identification is used to
create and maintain a map of the application layer
protocols present in the monitored network, and
generate a security event whenever a new protocol is
30   detected in violation of network policy. An advantage
of this method is that it provides network and security
administrators with an accurate, real time map of the
services deployed on the monitored network, regardless
of the port numbers used.
35   Further preferably, a network intrusion detection

9

or prevention system as described herein obtains information on the protocol of each observed data flow, either by making use of the protocol map above, or through online analysis of the data flow with the
5 protocol identification methods described above. This preferably leads to providing an intrusion detection or prevention system that makes use of the information on the application layer protocol involved in a given data flow to select the set of signatures and the set of
10 decoding, analysis, normalization, and reassembly tasks to be performed for the data flow. Matching misuse signatures against network traffic is a resource intensive task of ever growing complexity. Not only does the amount of computation and memory required for
15 this task grow with the bandwidth of the coupled network; this also grows with the number of misuse signatures employed, which in turn must grow as new technologies, protocols and applications are deployed on the network.

20   A basic advantage of the arrangement described herein lies in that, rather than enabling and disabling individual signatures for a given data flow, a signature set is selected based on the identified protocol. This allows the signature set relative to
25 each protocol to be preprocessed for efficient signature matching, without repeating the preprocessing each time a protocol is identified.

        Additionally, an IDS is thus provided that is capable of accurately detecting attacks regardless of
30 the port numbers involved in the communication, even in a fast changing network environment, reducing the amount of false negatives and false positives. Since the tasks performed by the IDS (and by the protocol identification system) are quite similar, both
35 involving the capture of network traffic, a number of

preprocessing tasks on the collected flow data, and the matching of the preprocessed flow data against a set of signatures, these two systems may easily and efficiently be integrated into a single system
5   performing both functions.

Brief description of the annexed drawings

The invention will now be described, by way of
10 example only, with reference to the enclosed figures of drawing, wherein:
    - Fig. 1 shows a typical layout of a network intrusion detection device in accordance with the arrangement described herein,
15   - Fig. 2 is a flow chart illustrative of operation of a preferred embodiment of the protocol identification system of Figure 1,
        - Fig. 3 is a flow chart illustrative of operation of a preferred embodiment of the protocol
20 identification with statistical characterization,
        - Fig. 4 is a flow chart illustrative of operation of a preferred embodiment of a network intrusion detection system as described herein, and
        - Fig. 5 is another flow chart illustrative of the
25 integration of a protocol identification feature in a network intrusion detection system.

Detailed description of preferred embodiments of the invention
30

    Figure 1 shows the typical layout of a protocol identification arrangement as described herein.

    A network, such as a local area network generally designated 2 and monitored by an intrusion detection
35 system 6, is connected to a public network 4, such as

the Internet, and is therefore open to access via a client device 8. A plurality of devices 10 are deployed on the network 2 for exchanging data and sharing resources, accessing the Internet 4, and offering
5   services to clients 8. Between the network 2 and the net 4 a firewall 12, shown in Figure 1 with a broken line, can be used for limiting external access to resources in the local area network 2 and protecting such resources from unauthorized use.

10   The intrusion detection system 6 is coupled to the monitored network 2 in order to detect and capture data being transmitted on the network.

The intrusion detection system 6 includes:

- a sniffer 14 for capturing data on the network
15   2,

- a protocol identification engine 16 which receives data captured by the sniffer 14, and

- an intrusion detection engine 18 which performs the intrusion detection proper by using information
20   obtained by the protocol identification engine 16.

The designation "sniffer" indicates a module that monitors network traffic and can be used to capture data being transmitted on a network. Thanks to the sniffer 14, the network intrusion detection system 6 is
25   able to read any packet of data passed on to the network 2, thus determining the source and destination addresses of the packet and analyzing the corresponding data content as explained in detail hereinbelow.

Figure 2 is a flow chart illustrating operation of
30   a preferred embodiment of the network intrusion detection system 6. In the exemplary embodiment shown, intrusion detection features are limited to detecting application layer protocols and verifying if said

12

protocols are in compliance with network policies.

In a step 19, the sniffer 14 captures traffic on the network 2. A subsequent preprocessing step 20 includes a plurality of preprocessing tasks on the

5   captured packet. The preprocessing tasks may include a plurality of decoding, analysis, reassembly and normalization tasks, such as IP de-fragmentation, TCP stream state tracking and reassembly, data flow analysis, protocol analysis and normalization. Each of

10   these tasks is, per se, well known in the art, thus making it unnecessary to provide a detailed description herein.

A signature-matching step 22, performed by a corresponding engine, leads to matching a preprocessed

15   data flow against a set of protocol detection signatures 24.

Signatures for protocol identification are specified in a signature language similar to the signature languages employed by conventional network

20   intrusion detection systems.

An example of such a language is the "snort" rule language, that is the language employed to specify detection signatures for the well known open source network intrusion detection system. As is well known,

25   "snort" is the de facto standard for the network ids (see e.g. www.snort.org). Signatures may specify tests such as comparison of specific header fields of the network packet against a value, search for a predetermined sequence of bytes in the packet payload,

30   as well as stateful tests, that is tests that require memory of specific features of the data flow previous to the packet currently being matched.

As an example, the state of a TCP connection may be tested, a sequence of bytes may be searched for in

reassembled stream data, or a specified event may be
required to have occurred a specified number of times
within a given interval.

A useful test that may be specified in a stateful
5   signature language specifies that a first signature is
to be enabled only as a consequence of the triggering
of a second signature, and only for a specified set of
packets (see again, by way of general reference, PCT
patent application 03WO-IT00505). A pattern matching
10 engine for searching attack signatures into data
packets is supported by a response analysis engine for
detecting response signatures into data packets sent
back from a monitored network/computer. When a suspect
signature (second signature) has been detected into a
15 packet, the system enters an alarm status starting a
monitoring process on the packets sent back from the
potentially   attacked   network/computer,   enabling
matching tests with the first signature. An alarm is
generated only in case the analysis of the response
20 packets produces as well a positive result. The
specified set of packets may be limited to the packets
in the same data flow as the packet triggering the
second signature, or may be further limited to the
first packet or application layer protocol data unit
25 captured in reply to the packet or application layer
protocol data unit that triggered the second signature.

In the context of protocol identification, such a
test may be employed to detect client request - server
reply behavior, with the advantage that the number of
30 false positives, that is the number of erroneous
protocol identification events, may be greatly reduced.

An example of the use of a stateful signature
language for protocol identification is discussed
below, for the detection of the FTP protocol. Going

14

back to Figure 2, in the step 26 a check is made as to whether the protocol identification engine has detected a protocol. If a protocol has been detected, processing proceeds towards a step 28 where a protocol map 30 is updated by inserting information on the newly detected protocol. The protocol map 30 is essentially a data base or data storage system holding information on the detected protocols. The protocol map is further discussed below.

Further in Figure 2, in a step 32 a policy engine compares the detected protocol in the protocol map 30 with a network policy 34. The network policy 34 is specified by the administrator of the protocol identification system. Such a network policy may assert that specific protocols are not allowed on the monitored network, that servers of specific protocols are not allowed on the monitored network, or that only specific hosts of the monitored network are allowed a server and/or client role in communications employing a given protocol on the monitored network.

Step 36 is essentially a test as to whether the detected protocol is in violation of network policy. If this is the case, processing evolves towards a step 38 where a security event is generated. As a consequence of the security event, operation similar to operation of a conventional intrusion detection system may occur, such as logging of the event, and alerting the administrator of the event.

As an improvement to the arrangement described above, protocol identification may be restricted to data flows involving a server in the monitored network and a client outside the monitored network. Under the assumption that the monitored network is protected by a firewall 12, this has the advantage that the

15

computational cost of protocol identification is
reduced. Additionally, the size and therefore memory
occupation of the protocol map 30 is reduced, by
limiting protocol identification to those security
5   critical services that are accessible through the
firewall 12 from outside the monitored network. A
further advantage is that this allows security
administrators to apply an extremely restrictive
network policy 34 to the protocol identification
10 system, while allowing a wide variety of protocols to
be employed within the monitored network.

Two preferred embodiments of the protocol map are
shown hereinbelow.

IP Address	IP Protocol	Service Port	Protocol
131.2.1.15	TCP	9999	ftp(21)
131.2.1.22	TCP	16100	http(80)

15

Client IP	Server IP	IP Protocol	Client Port	Server Port	Protocol
218.3.1.25	131.2.1.15	TCP	1712	9999	ftp(21)
145.12.1.22	131.2.1.22	TCP	10212	16100	http(80)
173.13.3.11	131.2.1.22	TCP	8310	16100	http(80)

All the values in the tables above are provided
only as examples of data that might be found in such
tables during operation.

20   The former table shows one preferred embodiment
of the protocol map. Such a protocol map is arranged
based on the IP addresses of hosts inside the monitored
network. For each IP address (column 1) in the

16

monitored network, information may be stored on which
service ports (column 3) of which IP protocol (column
2) are listening for which application layer protocol
(column 4).

5   An alternative embodiment is shown in the latter
table. The protocol map in that table is arranged based
on data flows involving hosts of the monitored network.
For each data flow observed on the monitored network,
identified by client IP address (column 1), server IP
10   address (column 2), IP protocol (column 3), client
service port (column 4) and server service port (column
5), information may be stored on which application
layer protocol is employed in the communication (column
6).

15   In the following some examples are provided of
stateful signatures for the detection of the ftp
protocol. These signatures identify a client request –
server reply behavior: specifically they recognize a
logon attempt, consisting, as specified by RFC 959, by
20   the ftp command

          USER <SP> <username> <CRLF>

          followed by a server response.

          The client request is matched by a signature of
25   the type:

          signature XX begin

               signature.message:"ftp client logon attempt"

               ip.destination_address: IN monitored network

30   ip.source_address: IN not trusted network

               ip.protocol:tcp

```
        tcp.session_state: from client

        tcp.payload.content: "USER", case_insensitive

        tcp.payload.content:"|0a|

        signature.action:trigger_response_signature :
5  XY,XZ

        end
```

The two server responses are matched by signatures
of the type:

10

```
        signature XY begin

        signature.message:"ftp service detected,  logon
without password"

        signature.type: response

15      signature.response.type: first reply packet in
tcp session

        ip.source_address: IN monitored network

        ip.destination_address: IN not trusted network

        ip.protocol:tcp

20      tcp.session_state: from server

        tcp.payload.content: "230", within 3

        tcp.payload.content:"|0a|

        signature.action:protocol_detected ftp 21

        end
```

25

```
        signature XZ begin

        signature.message:"ftp        service        detected,
password requested"
```

18

```
        signature.type: response

        signature.response.type: first reply packet in
tcp session

        ip.source_address: IN monitored network

        ip.destination_address: IN not trusted network

        ip.protocol:tcp

        tcp.session_state: from server

        tcp.payload.content: "331", within 3

        tcp.payload.content:"|0a|

        signature.action :protocol_detected ftp 21

        end
```

Several of the features of a stateful signature
language described above are used in this example.

The signature XX recognises a TCP data flow
containing the case insensitive string "user ", and a
carriage return:

```
        tcp.payload.content: "USER ", case_insensitive

        tcp.payload.content: "|0a|"
```

The data flow must be from the client side of the
tcp connection:

```
        tcp.session_state:  from client
```

The signature.action field in signature XX:

```
        signature.action:    trigger_response_signature:
XY, XZ
```

specifies that, as a consequence of a successful
match of signature XX, signatures XY and XZ are to be
activated.

Signatures XY and XZ each recognize a specific

server response code. The signature XY recognizes a 230 code, which signals a successful logon, while signature XZ recognizes a 331 code, which signals that a password is requested for logon. The strings "230" and "331" are
5   required to be found within the first 3 bytes of the payload of a TCP packet. In the signature XY:

       tcp.payload. content: "230", within 3

       tcp.payload. content: "|0a|"

       and in signature 48:

10   tcp.payload. content: "331", within 3

       tcp.payload. content: "|0a|"

In both signatures, it is specified that signature is a response signature, that matches only the first reply packet seen in a TCP session in response to the
15   packet that triggered the signature XX:

       signature.type: response

       signature.response.type: first reply packet in
tcp session

The applicant has verified that, by means of
20   verification signatures such as XX, XY and XZ a protocol identification system is capable of identifying the FTP protocol with practically no false positives and false negatives.

The protocol identification method just described
25   may be refined by means of flow characterization and classification of data flows. A limited set of features such as packet size, packet arrival times, tcp flags and other header information are gathered and statistics thereof are computed. The flow
30   classification engine then classifies the data flows and services into a number of flow classes. Interactive traffic can be discriminated from bulk traffic.

20

Additionally, a set of specific protocols, such as http and ssh, may be recognized with good accuracy without payload inspection.

The basic principles of flow classification performed without payload inspection are held to be known to those of skill in the art (as witnessed e.g. by Sun et al., "Statistical identification of encrypted web browsing traffic ", IEEE Symposium on Security and Privacy, 2002, or by Zhang et al., "Detecting backdoors", USENIX Security Symposium, ·2000). The information obtained by flow classification ·may be stored in a flow classification map similar to the latter protocol map considered in the foregoing, or in a service classification map similar to the former protocol considered in the foregoing map, where flow classification information are substituted to the protocol field of the tables discussed in the foregoing.

The flow or service classification map is correlated with the information gathered by signature based protocol identification to improve reliability of the identification. Furthermore, the signature language employed for protocol identification may be enriched to allow testing of flow and service classification information available in the flow and service classification map. As an example the following signature fragment is included:

        flow.type: interactive

Figure 3 ·is a flow chart of a specific embodiment of the arrangement considered here.

As in the case of the embodiment of Fig. 2, the sniffer 14 captures the network packets in the step 19, and the packets are analyzed in the preprocessing step 20. In a step 50, the flow and service feature

21

extraction engine updates a flow and service feature map 52 with information derived from the current packet. The feature map 52 is similar to the protocol maps considered in the foregoing and to a flow and service classification map 58, but it stores summary information on the traffic flow features needed for flow and service classification, such as statistics of packet size and of packet inter-arrival times. In a step 54 a flow and service classification engine analyses the feature map 52 and compares it with the flow classification knowledge base 56, to determine flow classification information for the current flow and/or service, which is stored in the flow and service classification map 58. In a step 60, a signature-matching engine then matches the current data flow against protocol identification signature set 62. The step 60 and the signature set 62 are expanded, with respect to the step 22 and the signature set 24 shown in Figure 2, to make use of the flow and service classification map 58.

The signatures 62 may specify flow classification requirements, and in the step 60 the engine matches such signatures by comparing flow classification requirements with information on the current flow or service in classification map 58.

Processing then evolves through steps 26 and 28 and further, as in Figure 2.

Figure 4 is a flow chart illustrative of operation of a network intrusion detection system making use of the protocol map 30 to improve reliability of misuse detection.

As in the protocol identification systems shown in Figure 2 and in Figure 3, and as in a conventional network intrusion detection system, in the step 19 the

sniffer 14 captures traffic on the network 2; this is subsequently processed in the step 20. In a step 64, a lookup is performed for the current packet in the protocol map 30, in order to determine the protocol involved in the communication.

If no information on the current service port or data flow is found in protocol map 30, a protocol is chosen as in a conventional IDS, based on source and destination service port numbers. Step 66 is representative of preprocessing specific for the selected protocol, which may perform a plurality of decoding, normalization, re-assembly and analysis tasks relative to the current protocol.

As an example for the http protocol, URLs may be normalized by removing non-standard character encodings, TCP stream reassembly may be performed on the client side of the connection but not on the server side to reduce computation and storage costs, and anomaly based security events may be generated if the http method is too long, which may be indication of a buffer overflow attempt. In the step 22 the signature matching engine performs matching of the current data flow against signature set 68, the misuse detection signature set for the selected protocol. Furthermore, in a step 70 a check is made as to whether a misuse was detected in the step 22, and, if that is the case, a security event is generated in a step 38.

Examination of Figures 2 and 4 shows that the protocol identification system described and a network intrusion detection system perform essentially similar processing on the traffic observed on the monitored network, and therefore can efficiently be integrated into a single system. A sniffer, a preprocessor, a signature-matching engine, and a security event

generation engine are present in both systems.

Figure 5 is a flow chart exemplary of operation od a network intrusion detection system with integrated protocol identification.

In Figure 5, processing proceeds through the sniffer 14, the preprocessor, through protocol selection in the step 64, and through the protocol specific preprocessing 66 as in the IDS illustrated in figure 4.

Further in Figure 5, in the step 22 the signature-matching engine performs matching of the current data flow against both the protocol identification signature set 24, and the misuse detection signature set for the selected protocol 68.

Although the signature sets 24 and 68 are shown separately in the diagram, they are not separately matched against the data flow by engine in the step 22; rather, the sets 24 and 68 are integrated into a single signature set, and in the step 22 the engine performs set-wise matching of the data flow against the entire signature set.

Again, in a block 26 a test is made whether a protocol was detected and, if it was, the protocol map 30 is updated in the step 28, while in the step 32 the policy engine compares the detected protocol with network policy 34. Further in the diagram, in a step 70 a check is made as to whether a misuse was detected, either by the signature matching engine (in the step 22), or by the policy engine (in the step 32). If that is the case, a security event is generated in the step 38.

Of course, without prejudice to the underlying principles of the invention, the details and the embodiments may vary, even significantly, with respect

to what has been described and shown by way of example only, without departing from the scope of invention as defined by the annexed claims.